

Large scale *ab initio* calculations based on three levels of parallelization

François Bottin^{a,b,*}, Stéphane Leroux^a, Andrew Knyazev^{c,1}, Gilles Zérah^a

^a*Département de Physique Théorique et Appliquée, CEA/DAM Ile-de-France, BP 12, 91680 Bruyères-le-Châtel Cedex, France*

^b*LRC - Centre de Mathématiques et de Leurs Applications, CNRS (UMR 8536)*

ENS Cachan, 61, Avenue du Président Wilson, Cachan Cedex, France

^c*Department of Mathematical Sciences, University of Colorado at Denver, P.O. Box 173364, Campus Box 170, Denver, CO 80217-3364.*

Received May 11, 2007; received in the revised form July 11, 2007; accepted July 22, 2007

Abstract

We suggest and implement a parallelization scheme based on an efficient multiband eigenvalue solver, called the locally optimal block preconditioned conjugate gradient (LOBPCG) method, and using an optimized three-dimensional (3D) fast Fourier transform (FFT) in the *ab initio* plane-wave code ABINIT. In addition to the standard data partitioning over processors corresponding to different \mathbf{k} -points, we introduce data partitioning with respect to blocks of bands as well as spatial partitioning in the Fourier space of coefficients over the plane waves basis set used in ABINIT. This \mathbf{k} -points-multiband-FFT parallelization avoids any collective communications on the whole set of processors relying instead on one-dimensional communications only. For a single \mathbf{k} -point, super-linear scaling is achieved for up to 100 processors due to an extensive use of hardware optimized BLAS, LAPACK and SCALAPACK routines, mainly in the LOBPCG routine. We observe good performance up to 200 processors. With 10 \mathbf{k} -points our three-way data partitioning results in linear scaling up to 1000 processors for a practical system used for testing.

©2007 Bottin, Leroux, Knyazev, Zérah. All rights reserved.

Key words: Density functional theory, ABINIT, Eigenvalue, LOBPCG, SCALAPACK, FFT, Parallelization, MPI, Supercomputing

PACS: 71.15.Dx

1. Introduction

The density functional theory (DFT) [1] and the Kohn-Sham (KS) equations [15] are efficient techniques reducing the costs of solving the original Schrödinger equations without sacrificing the accuracy in many practically important cases. However, numerical modeling of such properties as large surface reconstructions [2–5] and melting [6–9], which involves molecular dynamics and systems with hundreds atoms, remains too time consuming for single-processor computers even in the DFT-KS framework. Parallelization, i.e., software implementation suitable for com-

puters with many processors, of electronic structure codes has become one of the main software development tasks as tens of thousands processors are already available on massively parallel systems, see <http://www.top500.org>. In order to use efficiently these supercomputers, various *ab initio* codes are being deeply modified [10–14].

Efficient eigensolvers for iterative diagonalization of the Hamiltonian matrix [16] are necessary for large systems. In the framework of *ab initio* calculations, the most commonly used diagonalization method is the band-by-band conjugate gradient (CG) algorithm proposed by Teter *et al.* [17] for direct minimization of the total energy. Thereafter, it has been modified by Bylander *et al.* [18] to fit the iterative diagonalization framework. This algorithm is fast and robust for small matrices, but requires explicit orthogonalization of residual vectors to already resolved bands. For larger matrices, the residual minimization method—a direct inversion in the iterative subspace (RMM-DIIS) [19, 20]—is more efficient. This latter scheme based on the original work of Pulay [21] and modified by Wood and Zunger [22]

* Corresponding author. Tel.: +33 16926 4000

Email addresses: Francois.Bottin@cea.fr (François Bottin),

stephane.leroux@cea.fr (Stéphane Leroux),

andrew.knyazev@cudenver.edu (Andrew Knyazev),

gilles.zerah@cea.fr (Gilles Zérah).

URL: <http://math.cudenver.edu/~aknyazev/> (Andrew Knyazev).

¹ Partially supported by the NSF award DMS-0612751.

avoids orthogonalizations.

The popular preconditioned block Davidson scheme [23] is reported to be overtaken by the preconditioned CG (PCG) method for small and by the RMM-DIIS for large systems [20]. Other widely used methods for large systems are the Lanczos algorithm [24] with partial reorthogonalization and the Chebyshev-filtered subspace iterations [25] with one diagonalization at the first iteration, neither utilizes preconditioning.

In solid state physics, it is convenient to expand wave-functions, densities and potentials over a plane waves (PW) basis set, where the wave-functions identify themselves with Bloch's functions and periodic boundary conditions are naturally treated. Moreover, the dependency of the total energy functional on the PW basis set is variational in this case. Long-range potentials, which are hard to compute in real space, can be easily evaluated in reciprocal (Fourier) space of the PW expansion coefficients. Depending on the computational costs, quantities are computed in real or reciprocal space—and we go from one space to the other using backward and forward three-dimensional (3D) fast Fourier transformations (FFTs).

As iterative eigensolvers and FFTs dominate in the overall computational costs of PW-based electronic structure codes, such as ABINIT [26, 27], efficient coordinated parallelization is necessary. In the present paper, we suggest and implement a parallelization scheme based on an efficient multiband eigenvalue solver, called the locally optimal block preconditioned conjugate gradient (LOBPCG) method, and using optimized 3D FFTs in the *ab initio* PW code ABINIT. In addition to the standard data partitioning over processors corresponding to different \mathbf{k} -points, we introduce data partitioning with respect to blocks of bands as well as spatial partitioning in the Fourier space of PW coefficients.

The eigensolver that we use in this work is the LOBPCG method proposed by Knyazev [28–31]. Our choice of the LOBPCG is determined by its simplicity, effectiveness, ease of parallelization, acceptance in other application areas, and public availability of codes [32]. In the framework of *ab initio* self-consistent (SC) calculations, the use of LOBPCG is apparently novel, see also [33] where the LOBPCG is adapted for the total energy minimization.

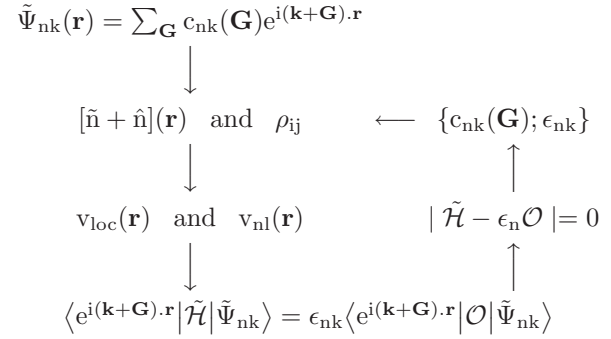
For FFTs, we use the 3D FFT implemented by Goedecker *et al.* [34], which has previously demonstrated its efficiency on massively parallel supercomputers. This algorithm is cache optimized, minimizes the number of collective communications, and allows FFTs with zero padding.

In section 2 we describe the flow chart and bottlenecks of self-consistent calculations to motivate our work on parallelization. We introduce the LOBPCG eigensolver and compare it to the standard CG eigensolver in section 3. Our multiband-FFT parallelization and the optimizations performed are described in section 4. In section 5 we demonstrate our numerical results and report scalability of our two-level (multiband-FFT) and three-level (\mathbf{k} -points-multiband-FFT) parallelization a large-size system.

2. Motivation to parallelize the self-consistent calculations

2.1. The self-consistent loop

In order to keep the problem setup general, we use the term in the right-hand side of the KS equations, the so-called overlap operator \mathcal{O} , which comes from the non-orthogonality of the wave-functions. It appears for ultra-soft pseudo-potentials (USPP) [35] or the projector augmented-wave (PAW) method [36] and is reduced to the identity in norm-conserving calculations. The minimum of the total energy functional is calculated by carrying on a SC determination of the density and potential in the SC loop. In the following flow-chart, we display schematically the SC loop with its various components in the framework of PAW calculations in ABINIT:



The initiation step is that the wave-function of the system $\Psi_{nk}(\mathbf{r})$, where n and k are the band and the \mathbf{k} -point indexes, respectively, is expanded over the PW basis set with coefficients $c_{nk}(\mathbf{G})$ and vectors \mathbf{G} in the reciprocal space. Now the SC loop starts: First, the wave-function is used to compute the pseudized charge density $\tilde{n}(\mathbf{r})$ as well as PAW specific quantities: the compensation charge $\hat{n}(\mathbf{r})$, needed to recover the correct multipolar development of the real charge density $n(\mathbf{r})$, and the matrix density ρ_{ij} , which defines the occupancies of the partial waves within the PAW spheres. Second, these densities are used to compute the local v_{loc} and non-local v_{nl} parts of the potential. Third, as the current Hamiltonian is determined, the linear (generalized if the overlap operator \mathcal{O} is present) eigenvalue problem projected over PW is now solved iteratively to obtain new approximate wave-functions PW coefficients $c_{nk}(\mathbf{G})$ corresponding to the minimal states. The next step is the subspace diagonalization involving all bands to improve the accuracy of approximate eigenvectors from the previous step. Finally, the input and output densities (or potentials) are mixed. The SC loop stops when the error is lower than a given tolerance.

2.2. Bottlenecks as motivation for parallelization

As the size of the system increases, various parts of the SC loop become very time consuming. Parallelization of

the most computationally expensive parts is expected to remove or at least widen the existing bottlenecks. We focus here on parallel algorithms to address the following tasks, in the order of importance: (i) iterative solution of linear generalized eigenvalue problems (step 3 of the SC loop); (ii) calculations of eigenvalues and eigenvectors in the subspace (step 4 of the SC loop); (iii) FFT routines where the local potential is applied to wave-functions or where the density is recomputed (necessary to apply the Hamiltonian in the eigensolver in step 3 of the SC loop); and (iv) computation of three non-local-like terms: the non-local part of the potential v_{nl} , the overlap operator \mathcal{O} , and the ρ_{ij} matrix (step 2 of the SC loop).

A brief review of our approaches to these problems follows: To solve approximately eigenproblems (i) in parallel, we use multiband (block) eigenvalue solvers, where energy minimization is iteratively performed in parallel for sets of bands combined into blocks. SCALAPACK is used to solve (ii). 3D FFTs are well suited to perform FFT in parallel to address (iii), distributing the calculations over processors by splitting the PW coefficients. Computation of non-local-like terms (iv) can be efficiently parallelized in the reciprocal space.

3. Multiband eigenvalue solver

3.1. Locally optimal eigenvalue solver

Our eigensolver, the LOBPCG method, is a CG-like algorithm, which in its single-band version [28] calls the Rayleigh-Ritz procedure to minimize the eigenvalue ϵ within the 3D subspace Ξ spanned by $\{\psi^{(i)}, \nabla\epsilon(\psi^{(i)}), \psi^{(i-1)}\}$, where the index i represents the iteration step number during the minimization and $\nabla\epsilon(\psi^{(i)})$ is the gradient of the Rayleigh quotient $\epsilon(\psi^{(i)})$ given by

$$\epsilon(\psi) = \frac{\langle \psi | \mathcal{H} | \psi \rangle}{\langle \psi | \mathcal{O} | \psi \rangle}, \quad \nabla\epsilon(\psi) = 2 \frac{\mathcal{H}\psi - \epsilon(\psi)\mathcal{O}\psi}{\langle \psi | \mathcal{O} | \psi \rangle}. \quad (1)$$

Since scaling of basis vectors is irrelevant, we replace $\nabla\epsilon(\psi^{(i)})$ with $\mathbf{r}^{(i)} = \mathcal{H}\psi^{(i)} - \epsilon^{(i)}\mathcal{O}\psi^{(i)}$, so the new Ritz vector is $\psi^{(i+1)} = \delta^{(i)}\psi^{(i)} + \lambda^{(i)}\mathbf{r}^{(i)} + \gamma^{(i)}\psi^{(i-1)}$, with the scalar coefficients $\delta^{(i)}$, $\lambda^{(i)}$ and $\gamma^{(i)}$ being obtained by the Rayleigh-Ritz minimization on the subspace Ξ . The use of the variational principal to compute the iterative parameters justifies the name “locally optimal” and makes this method distinctive, as in other PCG methods, e.g., in [17], formulas for iterative parameters are explicit.

To avoid the instability in the Rayleigh-Ritz procedure arising when $\psi^{(i)}$ becomes close to $\psi^{(i-1)}$ as the method converges toward the minimum, a new basis vector $\mathbf{p}^{(i+1)} = \lambda^{(i)}\mathbf{r}^{(i)} + \gamma^{(i)}\mathbf{p}^{(i)} = \psi^{(i+1)} - \delta^{(i)}\psi^{(i)}$, where $\mathbf{p}^{(0)} = 0$, has been introduced [31]. Replacing $\psi^{(i-1)}$ with $\mathbf{p}^{(i)}$ in the basis of the minimization subspace Ξ does not change it but gives a more numerically stable algorithm:

$$\psi^{(i+1)} = \delta^{(i)}\psi^{(i)} + \lambda^{(i)}\mathbf{r}^{(i)} + \gamma^{(i)}\mathbf{p}^{(i)}. \quad (2)$$

In order to accelerate the convergence and thus to improve the performance, a PW optimized preconditioner K is introduced and the preconditioned residual vectors $\mathbf{w}^{(i)} = K(\mathbf{r}^{(i)}) = K(\mathcal{H}\psi^{(i)} - \epsilon^{(i)}\mathcal{O}\psi^{(i)})$ replace $\mathbf{r}^{(i)}$ in (2). In the next subsection we describe a multiband, or block, version of the LOBPCG method, which is especially suitable for parallel computations.

3.2. Locally optimal multiband solver

The single-band method (2) can be used in the standard band-by-band mode, where the currently iterated band is constrained to be \mathcal{O} -orthogonal to the previously computed ones. Alternatively, method (2) can be easily generalized [29–31] to iterate a block of $m > 1$ vectors to compute m bands simultaneously. It requires diagonalization of a matrix of the size $3m$ on every iteration.

In large-scale *ab initio* calculations the total number M of bands is too big to fit all M bands into one block $m = M$ as the computational costs of the Rayleigh-Ritz procedure on the $3m$ dimensional subspace Ξ becomes prohibitive. Instead, M eigenvectors and eigenvalues are split into blocks $\Psi = \{\psi_1, \dots, \psi_m\}$ and $\Upsilon = \text{diag}\{\epsilon_1, \dots, \epsilon_m\}$ of the size $m < M$. The other quantities involved in this algorithm are also defined by blocks; the small letters are then replaced by capital letters, e.g., the scalars $\delta^{(i)}$, $\lambda^{(i)}$ and $\gamma^{(i)}$ are replaced with m -by- m matrices $\Delta^{(i)}$, $\Lambda^{(i)}$ and $\Gamma^{(i)}$. The algorithm for the block-vector Ψ of m bands that we use here is the same as that in the LOBPCG code in the general purpose library BLOPEX [32]:

Require: Let $\Psi^{(0)}$ be a block-vector and K be a preconditioner; the $\mathbf{P}^{(0)}$ is initialized to 0.

for $i=1, \dots, \text{nline}$ **do**

(i) $\Upsilon^{(i)} = \Upsilon(\Psi^{(i)})$

(ii) $\mathbf{R}^{(i)} = \mathcal{H}\Psi^{(i)} - \mathcal{O}\Psi^{(i)}\Upsilon^{(i)}$

(iii) $\mathbf{W}^{(i)} = K(\mathbf{R}^{(i)})$

(iv) We apply the Rayleigh-Ritz method within the subspace Ξ spanned by the columns of $\Psi^{(i)}$, $\mathbf{W}^{(i)}$ and $\mathbf{P}^{(i)}$ to find $\Psi^{(i+1)} = \Psi^{(i)}\Delta^{(i)} + \mathbf{W}^{(i)}\Lambda^{(i)} + \mathbf{P}^{(i)}\Gamma^{(i)}$ corresponding to the minimal m states within Ξ

(v) $\mathbf{P}^{(i+1)} = \mathbf{W}^{(i)}\Lambda^{(i)} + \mathbf{P}^{(i)}\Gamma^{(i)}$

end for

Algorithm 1. The locally optimal multiband solver LOBPCG

The LOBPCG algorithm as formulated above computes only the m lowest states, while we need to determine all $M > m$ states. So we perform LOBPCG algorithm within the loop over the blocks, where the current LOBPCG block is constrained to be \mathcal{O} -orthogonal to the previous ones. In other words, we iterate in a multiband-by-multiband fashion. Typically we perform a fixed number **nline** of iterations in each sweep on the eigensolver, so the total number of iterations to approximate the solution to the eigenvalue problem with the given fixed Hamiltonian is **nline** M/m .

To improve the accuracy, we conclude these multiband-by-multiband iterations with a final Rayleigh-Ritz call for all M bands, as pointed out in the description of the SC loop in section 2, but this procedure is technically outside of the LOBPCG routine in the ABINIT code.

3.3. LOBPCG vs. the standard PCG

The LOBPCG algorithm is much better supported theoretically [37] and can even outperform the traditional PCG method [17] as we now demonstrate. We test band-by-band LOBPCG, full-band LOBPCG and CG methods for two different systems: a simple one with two atoms of carbon within a diamond structure and a somewhat larger and rather complex system with sixteen atoms of plutonium in the α monoclinic structure.

The time per iteration is approximately the same for the single-band LOBPCG and PCG methods for single-processor calculations, so the complexity of one self-consistent electronic step is roughly the same for both methods. For the full-band LOBPCG the number of floating-point operations is higher than that in the band-by-band version because of the need to perform the Rayleigh-Ritz procedure on the $3m = 3M$ dimensional subspace, but the extensive use of the high-level BLAS for matrix-matrix operations in LOBPCG may compensate for this increase. Thus, we display the variation of the total energy as a function of the number of self-consistent electronic steps and, as the efficiency criterion, we use the final number of these steps needed for the variation of the total energy to reach the given tolerance in the SC loop. The numbers m and M are called **blocksize**

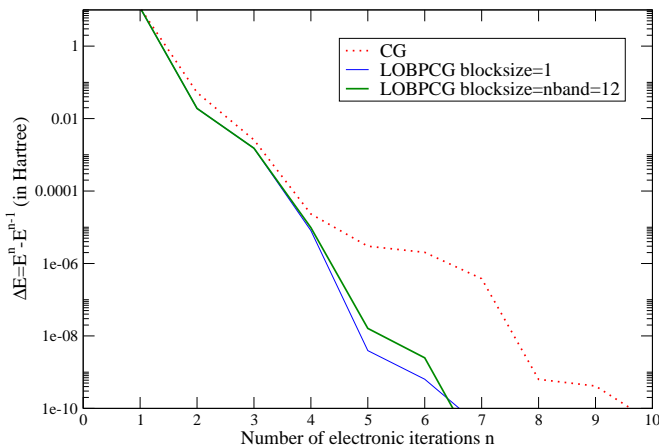


Fig. 1. Total energy variation of carbon in its diamond phase as a function of the self-consistent steps for band-by-band PCG and ($m = \text{blocksize} = 1$) LOBPCG methods, and full-band ($m = \text{blocksize} = M = \text{nband} = 12$) LOBPCG method. The index n stands for the number of electronic iterations within the SC loop, and E^n denotes the total energy at the n^{th} iteration

We show the variation of the total energy as a function of the number of electronic steps for the simple system using

the tolerance 10^{-10} Hartree in Figure 1. In this example, the number of iterations **nline** carried out by the LOBPCG or PCG method at each electronic step is set to 4, which is a typical value in ABINIT for many systems. Increasing **nline** does not change the number of electronic steps needed to reach the 10^{-10} Hartree tolerance in this case and therefore is inefficient. Convergence is slightly faster for the LOBPCG method, which takes 7 electronic steps vs. 10 steps needed by the PCG. The full-band LOBPCG method does not im-

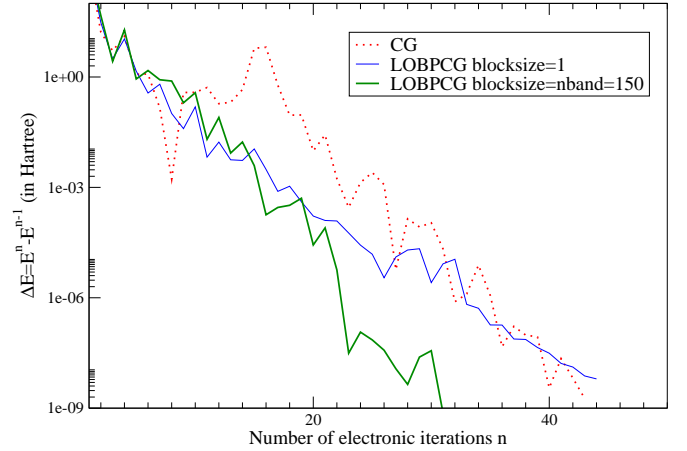


Fig. 2. Same caption as Fig. 1 for the plutonium in its α phase with $M = \text{nband} = 150$.

For the plutonium in its α phase system, the typical value **nline**=4 appears too small, so we increase it to **nline**=6 and display the results in Figure 2. We observe that the number of electronic steps is 40 for both band-by-band LOBPCG and PCG methods. But the full-band ($m = \text{blocksize} = M = \text{nband} = 150$) LOBPCG method finishes first with a clear lead in just 30 electronic steps. Further increase of **nline** does not noticeably affect the number of electronic steps.

We conclude that the LOBPCG method is competitive compared to the standard PCG method and that choosing a larger block size $m > 1$ in LOBPCG may help to further improve the performance in sequential (one-processor) computations. However, the main advantage of the LOBPCG method is that it can be efficiently parallelized as we describe in the next section.

4. The band-FFT parallelization

4.1. Distribution of the data

The LOBPCG algorithm works multiband-by-multiband, or block-by-block, so it can be easily parallelized over bands by distributing the coefficients of the block over the processors. The main challenge is to make the multiband data partitioning compatible with the data partitioning suitable for the 3D parallel FFT. In this section we describe these

partitioning schemes and an efficient transformation between them.

We implement two levels of parallelization using virtual Cartesian two dimensional MPI topology with the so-called “FFT-processors” and “band-processors” along virtual horizontal and vertical directions, correspondingly. Let P be the number of planes along one direction of the 3D reciprocal lattice of vectors $\{\mathbf{G}\}$. We choose numbers m and p of band- and FFT-processors to be divisors of M and P , respectively, so that the total number of processors is $n_{\text{proc}} = p \times m$. All the quantities expanded over the PW basis set: wave-functions, densities, potentials, etc., undergo similar distributions. We focus below on the coefficients of the wave-functions, and make a comment about other data at the end of the section.

We omit the index k in the following. We assume that the PW coefficients $c_n(\mathbf{G})$ of a given band n are originally defined on the processor at the virtual corner and first the P planes are successively distributed over the p FFT-processors, such that the PW coefficients of all the bands are distributed along the first row of the virtual two dimensional processor topology:

$$m \left\{ \begin{array}{c} \overbrace{\begin{pmatrix} c_n(\mathbf{G}) & - & - & - \\ - & - & - & - \\ - & - & - & - \\ - & - & - & - \end{pmatrix}}^p \\ \end{array} \right\} \rightarrow \left\{ \begin{array}{cccc} c_n(\mathbf{G}_1) & c_n(\mathbf{G}_2) & \dots & c_n(\mathbf{G}_p) \\ - & - & - & - \\ - & - & - & - \\ - & - & - & - \end{array} \right\}$$

There are now P/p planes defined on each processor. The $\{\mathbf{G}_i\}$ sub-sets for $i = 1, 2, \dots, p$ define a partition of the whole 3D reciprocal lattice of vectors $\{\mathbf{G}\}$. This distribution is typically used in conjunction with Goedecker’s parallel FFT routine. To parallelize further, these planes are sub-divided again by distributing the PW coefficients over the m band-processors as:

$$\left\{ \begin{array}{cccc} c_n(\mathbf{G}_{11}) & c_n(\mathbf{G}_{21}) & \dots & c_n(\mathbf{G}_{p1}) \\ c_n(\mathbf{G}_{12}) & c_n(\mathbf{G}_{22}) & \dots & c_n(\mathbf{G}_{p2}) \\ \vdots & \vdots & \ddots & \vdots \\ c_n(\mathbf{G}_{1m}) & \dots & \dots & c_n(\mathbf{G}_{pm}) \end{array} \right\}$$

The $\{\mathbf{G}_{ij}\}$ sub-sub-sets for $j = 1, 2, \dots, p$ partition the $\{\mathbf{G}_i\}$ sub-set. When performing LOBPCG calculations with blocks of size m , the coefficients are grouped in M/m blocks of size m , e.g., for the first m bands we have

$$\left\{ \begin{array}{cccc} c_{1:m}(\mathbf{G}_{11}) & c_{1:m}(\mathbf{G}_{21}) & \dots & c_{1:m}(\mathbf{G}_{p1}) \\ c_{1:m}(\mathbf{G}_{12}) & c_{1:m}(\mathbf{G}_{22}) & \dots & c_{1:m}(\mathbf{G}_{p2}) \\ \vdots & \vdots & \ddots & \vdots \\ c_{1:m}(\mathbf{G}_{1m}) & \dots & \dots & c_{1:m}(\mathbf{G}_{pm}) \end{array} \right\}$$

This data distribution is well suited to perform dot-products as well as matrix-vector or matrix-matrix operations needed in the LOBPCG algorithm. In particular, blocks of the Gram matrix are computed by the BLAS function `zgemm` on each processor using this distribution. In the next section we will show that the `zgemm` function plays the main role in superlinear scaling of the LOBPCG algorithm.

However, such a distribution is not appropriate to perform parallel 3D FFTs and thus has to be modified. This is achieved by transposing the PW coefficients inside each column, so the following distribution of the first m bands is thus obtained:

$$\left\{ \begin{array}{cccc} c_1(\mathbf{G}_1) & c_1(\mathbf{G}_2) & \dots & c_1(\mathbf{G}_p) \\ c_2(\mathbf{G}_1) & c_2(\mathbf{G}_2) & \dots & c_2(\mathbf{G}_p) \\ \vdots & \vdots & \ddots & \vdots \\ c_m(\mathbf{G}_1) & \dots & \dots & c_m(\mathbf{G}_p) \end{array} \right\}$$

where each band is distributed over the p FFT-processors. The distribution is now suitable to perform 3D parallel FFTs. The efficient 3D FFT of Goedecker *et al.* [34] implemented in our code is then applied on each virtual row of the processor partition, so m parallel 3D FFTs are performed simultaneously.

The transformation used to transpose the coefficients within a column corresponds to the `MPI_ALLTOALL` communication using the “band communicator.” During the 3D FFT, the `MPI_ALLTOALL` communication is also performed within each row using the “FFT communicator.” These two communications are not global, i.e., they do not involve communications between all processors, but rather they are local, involving processors only within virtual rows or columns. After `nline` LOBPCG iterations for this first block of bands, we apply the same strategy to the next block, with the constraint that all bands in the block are orthogonal to the bands previously computed.

Finally, this last distribution is used to compute by FFT the contributions of a given band to the density, which are subsequently summed up over all bands-processors to calculate the electronic density distributed over the FFT-processors. The electronic density is in its turn used to produce local potentials having the same data distribution.

4.2. Subspace diagonalization

Here we address the issue (ii) brought up in section 2 that diagonalization of a matrix of the size $M \gg 1$ is computationally expensive. The standard implementation of this calculation is performed in the sequential mode by calling LAPACK to diagonalize the same matrix on every processor. It becomes inefficient if the number of bands M or the number of processors increases. To remove this bottleneck, we call the SCALAPACK library, which is a parallel version of LAPACK. The tests performed using SCALAPACK show that this bottleneck is essentially removed for systems up

to 2000 bands. For instance, on 216 processors this diagonalization takes 23% of the total time for a system with 1296 bands using LAPACK on each processor, but only 3% using SCALAPACK.

4.3. Generalization of the transposition principle

Finally, we address bottleneck (iv) formulated in section 2, i.e., computation of the three non-local like terms (the non-local part of the potential v_{nl} , the overlap operator \mathcal{O} and the ρ_{ij} matrix), which is one of the most time consuming parts of the SC loop. Parallelization is straightforward if these terms are computed in reciprocal space and the distribution of the PW is efficient. The data distribution here corresponds to the one used in the LOBPCG.

Computation of these terms can be made more efficient in parallel if we utilize the data distribution used for the 3D parallel FFTs based on one virtual line of processors [11, 12]. The cause of better efficiency here comes from merging two parallel features. On the one hand, we avoid any collective communication on the whole set of processors, thus reducing the data over each line independently rather than over the whole set of processors. On the other hand, we obtain better load-balancing of the processors since the FFT data are not sub-divided in this case.

5. Numerical Results

Calculations are performed on Tera-10 NovaScale 5160 supercomputer, composed of 544 nodes (with 8 dual-core 1.6 GHz Intel Montecito processors per node) connected by Quadrics at the “Commissariat à l’énergie atomique.” Benchmarks are carried out on a supercell calculation with 108 atoms of gold set in their Face Centered Cubic lattice positions and 648 bands. An energy cutoff of 24 Ha is used in these calculations, leading to a three dimensional reciprocal grid with 108^3 points. In all tests $nline=4$.

5.1. The two-level parallelization

Calculations are performed on various numbers of processors $nproc=1, 4, 18, 54, 108, 162$, and 216. For each value of $nproc$, we consider all distributions $m \times p$ which are allowed within this framework. We remind the reader that p has to be smaller than $108/2=54$, in order to use the 3D parallel FFT, whereas m and p have to be divisors of $M = 648$ and $P = 108$, respectively. For instance, for $nproc=18$, we can choose $m \times p = 18 \times 1, 9 \times 2, 6 \times 3, 3 \times 6, 2 \times 9$ and 1×18 .

We focus on three kinds of benchmarks here: the first two are $m \times 1$ (band-only) and $1 \times p$ (FFT-only) distributions. The virtual MPI topology is one dimensional in these cases. The third kind is the optimized $m \times p$ (band-FFT) distribution. The speedups for these three parallelization schemes are shown in Figure 3.

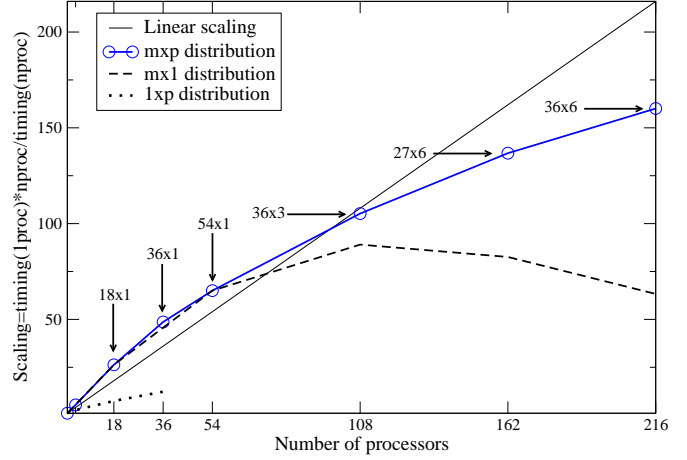


Fig. 3. Two-level parallelization: Speedup of the ABINIT code with respect to the number of the processors for $m \times p$, $m \times 1$, and $1 \times p$ data distributions. The “ideal” linear speedup is displayed by a line with the slope 1. For each arrow, we give numbers m and p of band- and FFT-processors, correspondingly.

The transposition of the data distribution within each column during the computation of v_{nl} and the use of collective communications inside lines for the FFT, applying v_{loc} to the wave-functions, are the two most important communications used in the framework of the Band-FFT parallelization. The communications within the FFTs increase to 17% of the total time on 216 processors using the $1 \times p$ (FFT-only). In contrast, the $m \times 1$ (band-only) distribution is adequate for up to 54 processors. However, further increase of the block size m together with the number of processors is inefficient in this case for more than 100 processors. We remind the reader that the LOBPCG solver performs the Rayleigh-Ritz procedure on subspaces of dimension $3m$ on every iteration. For large m this starts dominating the computational costs even on m processors. So we reduce m and introduce our double parallelization. For the optimal band-FFT parallelization we obtain a superlinear scaling up to 100 processors, and a speedup of 150 for 200 processors.

Figure 4 displays scaling of different parts of the ABINIT code using a slightly different format: the vertical axis represents the scalability divided by the number of processors, so the linear scaling is now displayed by the horizontal line. We also zoom in the horizontal axis to better represent the scalability for small numbers of processors. The ABINIT curve shows exactly the same data as in Figure 3. The LOBPCG curve represents the cumulative timing for the main parts that the LOBPCG function calls in ABINIT: $zgemm$, non-local (v_{nl}) and local (v_{loc} using FFT) parts of the potential.

Figure 4 demonstrates that the super-linear behavior is due to the $zgemm$. In the sequential mode we run the band-by-band eigensolver that does not take much advantage of the BLAS function $zgemm$ use, since only matrix-vector products are performed in this case. So the super-linear behavior is not directly related to the number of processors, but is rather explained by the known positive effect of vec-

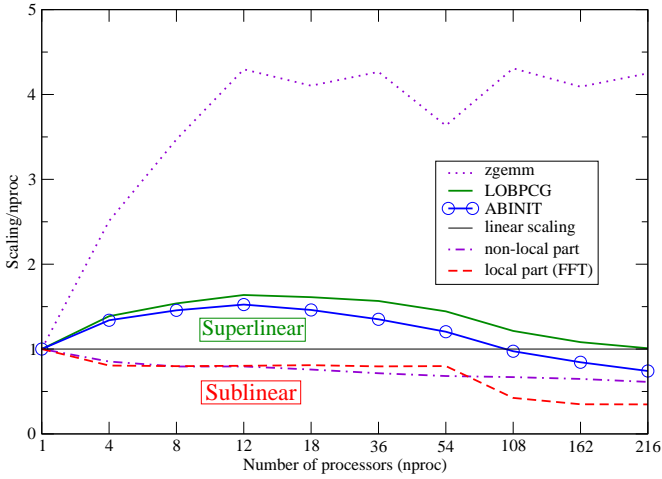


Fig. 4. Two-level parallelization: Speedup of various parts of the SC loop using the same $m \times p$ distribution as the best (top) curve in Figure 3. The main parts displayed are: **zgemm**, non-local (v_{nl}) and local (v_{loc} using FFT) parts of the potential. The LOBPCG represents mainly the sum of the above. The ABINIT curve is the same as on Figure 3.

tor blocking in LOBPCG that allows calling more efficient BLAS level 3 functions for matrix-matrix operation rather than BLAS level 2 functions for matrix-vector operations. This effect is especially noticeable in LOBPCG since it is intentionally written in such a way that the vast majority of linear algebra related operations are matrix-matrix operations (performed in LOBPCG using the **zgemm**), which represent 60% of the total time in band-by-band calculations on a single processor, but exhibits a strong speedup with the increase of the block size in the parallel mode.

The acceleration effect is hardware-dependent and is a result on such subtle processor features as multi-level cache, pipelining, and internal parallelism. Using the hardware-optimized BLAS is crucial: in one of our tests the time spent in the **zgemm** routine represents only 10% of the total time using block size $m = nproc = 54$ if hardware-optimized BLAS is used, but 53% otherwise. All the computation results reported here are obtained by running the ABINIT code with the hardware-optimized BLAS.

The effectiveness of the BLAS level 3 matrix-matrix **zgemm** also depends on the sizes of the matrices involved in the computations. This is the point where the parallel implementation that partitions the matrices may enter the scene and play its role in the acceleration effect.

5.2. The three-level parallelization

Parallelization over **k**-points is commonly used in *ab initio* calculations and is also available in ABINIT. It is very efficient and easy to implement, and results in a roughly linear scaling with respect to the number of processors. This kind of parallelization is evidently limited by the number k of **k**-points available in the system. It appears theoretically useless for large supercell calculations or disorder systems with only the Γ point within the Brillouin zone, but in prac-

tice sampling of the Brillouin zone is sometimes needed, e.g., for surfaces and interfaces in the direction perpendicular to the stacking sequence [4, 5], or for accurate thermodynamic integrations for metaling [8, 9]. Parallelization over **k**-points is clearly well suited for metals where a large number k of **k**-points is required.

We now combine the **k**-point and the Band-FFT parallelization by constructing a virtual three-dimensional configuration of processors $m \times p \times k$, where the Band-FFT parallelization is applied on each **k**-point in parallel.

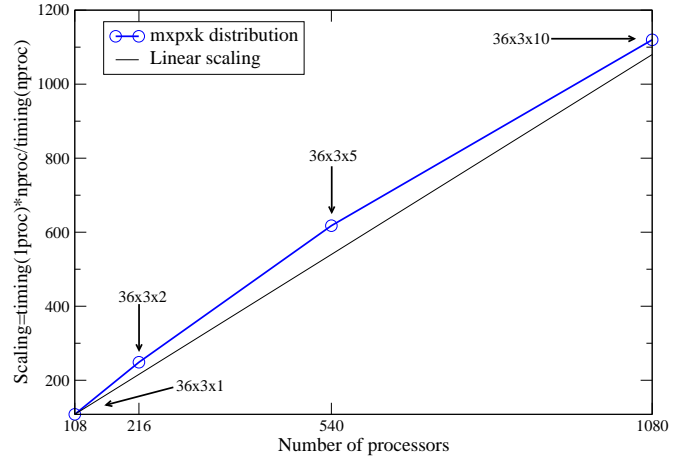


Fig. 5. The triple parallelization: speedup of the ABINIT code with respect to the number of processors.

In Figure 5 we present the speedup obtained for this triple parallelization. We use the same system as previously except that 10 **k**-points rather than 1 are sampled in the Brillouin zone. We use as the reference the limit of the superlinear scaling obtained in the framework of the Band-FFT parallelization: 108 processors with a 36×3 distribution. As expected, adding **k**-point processors in the virtual third dimension of the 3D processor configuration results in the linear scaling of computational costs. Using our three-level parallelization on 10 **k**-points the ABINIT scales linearly up to 1000 processors.

6. Conclusion

Our combined **k**-points-multiband-FFT parallelization allows the material science community to perform large scale electronic structure calculations efficiently on massively parallel supercomputers up to thousand of processors using the publicly available software ABINIT. The next generation of petascale supercomputers will bring new challenges. Novel numerical algorithms and advances in computer sciences, e.g., introduction of asynchronous (non-blocking) collective communication standards [38], would help us to minimize the communication load in future versions of ABINIT and make efficient use of the next generation hardware.

Acknowledgements The authors thank A. Curioni for fruitful discussions about the double parallelization technique and T. Hoefler about 3D FFT in parallel.

References

- [1] P. Hohenberg, W. Kohn, Phys. Rev. 136 (1964) B864.
- [2] K. D. Brommer, M. Needels, B. Larson, J. D. Joannopoulos, Phys. Rev. Lett. 68 (1992) 1355.
- [3] O. Dulub, U. Diebold, G. Kresse, Phys. Rev. Lett. 90 (2003) 016102.
- [4] G. Kresse, O. Dulub, U. Diebold, Phys. Rev. B 68 (2003) 245409.
- [5] D. Segeva, C. G. V. de Walle, Surf. Sci. 601 (2007) L15.
- [6] T. Ogitsu, E. Schwegler, F. Gygi, G. Galli, Phys. Rev. Lett. 91 (2003) 175502.
- [7] O. Sugino, R. Car, Phys. Rev. Lett. 74 (1995) 1823.
- [8] X. Wang, S. Scandolo, R. Car, Phys. Rev. Lett. 95 (2005) 185701.
- [9] F. Cricchio, A. B. Belonoshko, L. Burakovsky, D. L. Preston, R. Ahuja, Phys. Rev. B 73 (2006) 140103(R).
- [10] R. V. Vadali, Y. Shi, S. Kumar, L. V. Kale, M. E. Tuckerman, G. J. Martyna, J. Comp. Chem. 25 (2004) 2006.
- [11] J. Hutter, A. Curioni, Chem. Phys. Chem. 6 (2005) 1788.
- [12] J. Hutter, A. Curioni, Parallel Computing 31 (2005) 1.
- [13] C. K. Skylaris, P. Haynes, A. A. Mostofi, M. C. Payne, Phys. Stat. Sol. 243 (2006) 973.
- [14] C. K. Skylaris, P. Haynes, A. A. Mostofi, M. C. Payne, J. Chem. Phys. 122 (2005) 084119.
- [15] W. Kohn, L. J. Sham, Phys. Rev. 140 (1965) A1133.
- [16] M. Payne, M. Teter, D. Allan, T. Arias, J. Joannopoulos, Rev. of Mod. Phys. 64 (1992) 1045.
- [17] M. P. Teter, M. C. Payne, D. C. Allan, Phys. Rev. B 40 (1989) 12255.
- [18] D. M. Bylander, L. Kleinman, S. Lee, Phys. Rev. B 42 (1990) 1394.
- [19] G. Kresse, J. Furthmüller, Phys. Rev. B 54 (1996) 11169.
- [20] G. Kresse, J. Furthmüller, Comput. Mat. Sci. 6 (1996) 15.
- [21] P. Pulay, Chem. Phys. Lett. 73 (1980) 393.
- [22] D. M. Wood, A. Zunger, J. Phys. A: Math. Gen. 18 (1985) 1343.
- [23] E. R. Davidson, J. Comput. Phys. 17 (1975) 87.
- [24] C. Lanczos, J. Res. Nat. Bur. Standards 45 (1950) 255.
- [25] Y. Zhou, Y. Saad, M. L. Tiago, J. R. Chelikowsky, Phys. Rev. E 74 (2006) 066704.
- [26] ABINIT code, a joint project of the Université Catholique de Louvain, Corning Incorporated, and other contributors (URL <http://www.abinit.org>).
- [27] X. Gonze, J.-M. Beuken, R. Caracas, F. Detraux, M. Fuchs, G.-M. Rignanese, L. Sindic, M. Verstraete, G. Zerah, F. Jollet, M. Torrent, A. Roy, M. Mikami, P. Ghosez, J.-Y. Raty, D. Allan, Comput. Mater. Sci. 25 (2002) 478.
- [28] A. V. Knyazev, In International Ser. Numerical Mathematics, v. 96, Birkhauser, Basel, 1991, pp. 143–154.
- [29] A. V. Knyazev, Electron. Trans. Numer. Anal. 7 (1998) 104–123.
- [30] A. V. Knyazev, In Z. Bai et al. (Eds.), Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, SIAM, Philadelphia, 2000, pp. 352–368, section 11.3.
- [31] A. Knyazev, SIAM J. Sci. Computing 23 (2001) 517.
- [32] A. V. Knyazev, M. E. Argentati, I. Lashuk, E. E. Ovtchinnikov, SIAM J. Sci. Computing. Accepted. Published at <http://arxiv.org/abs/0705.2626>.
- [33] C. Yang, J. Meza, L. Wang, J. Comp. Physics 217 (2006) 709–721.
- [34] S. Goedecker, M. Boulet, T. Deutsch, Comput. Phys. Comm. 154 (2003) 105.
- [35] D. Vanderbilt, Phys. Rev. B 41 (1990) 7892.
- [36] P. E. Blöchl, Phys. Rev. B 50 (1994) 17953.
- [37] A. V. Knyazev, K. Neymeyr, Linear Algebra Appl. 358 (1-3) (2003) 95–114.
- [38] T. Hoefler, J. Squyres, G. Bosilca, G. Fagg, A. Lumsdaine, W. Rehm, www.cs.indiana.edu/pub/techreports/TR636.pdf